Dynamic Interior Point Method for Vehicular Traffic Optimization

Chang Guo, Demin Li, Guanglin Zhang, Xiaoning Ding, Reza Curtmola, and Cristian Borcea, *Member, IEEE*

***Abstract*—The aim of this article is to improve vehicular traffic in terms of both travel time and load balance. To achieve this goal, we propose an optimization model that minimizes the sum of the total travel time in the road network and a time representation of the traffic imbalance effects in the network. This paper presents an analytic formulation of the optimization problem, and an algorithm, Dynamic Interior Point Method (DIPM), that solves this optimization through driver rerouting. Unlike user-optimum traffic optimizations, DIPM leads to better fairness for drivers and works well in case of congestion. Unlike other system-wide traffic optimizations, DIPM considers the effects of the driver behavior on traffic load. Together, these features allow our system to work well in a potential real-world deployment. DIPM benefits from a central server that computes driver routes, which is reachable via cellular networks or vehicular ad hoc networks. Theoretical analysis and simulation results demonstrate that DIPM is fast and can work in real-time. The results of extensive simulations with realistic urban maps and traffic scenarios show that DIPM outperforms other dynamic rerouting algorithms in terms of travel time. DIPM also improves fairness when compared with a user-optimum approach.**

***Index Terms*—Travel time optimization, traffic load balance, vehicular networking, dynamic interior point method.**

I. Introduction

WITH the rapid mobility increase in urban areas worldwide, traffic congestion has become an urgent problem that requires a rapid and effective solution [1]. The Global Mobility Report in 2017 showed an additional 1.2 billion cars on the road compared to 2015's [2]. Americans waste nearly 14.5 million hours every day in traffic congestion [15]. Therefore, optimal route planning, which reduces the drivers' travel time and relieves congestion, has attracted much attention from academia and traffic management organizations [3].

Manuscript received February 14, 2019; revised October 11, 2019 and January 1, 2020; accepted March 16, 2020. Date of publication March 31, 2020; date of current version May 14, 2020. This work was supported in part by the NSF of China under Grants 71171045 and 61772130, in part by the Fundamental Research Funds for the Central Universities No. 2232020A-12, in part by the Innovation Program of Shanghai Municipal Education Commission under Grant 14YZ130, in part by the International S&T Cooperation Program of Shanghai Science and Technology Commission under Grant 15220710600, in part by the Fundamental Research Funds for the Central Universities No.17D310404, and in part by the U.S. National Science Foundation under Grants DGE 1565478, SHF 1617749, and CNS 1801430. The review of this article was coordinated by Dr. A. Heydari. *(Corresponding author: Guanglin Zhang.)*

Chang Guo, Demin Li, and Guanglin Zhang are with the College of Information Science and Technology, and Engineering Research Center of Digitized Textile and Apparel Technology, Ministry of Education, Donghua University, Shanghai 201620, China (e-mail: guochang@mail.dhu.edu.cn; deminli@dhu.edu.cn; glzhang@dhu.edu.cn).

Xiaoning Ding, Reza Curtmola, and Cristian Borcea are with the Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102-1982 USA (e-mail: xiaoning.ding@njit.edu; reza.curtmola@njit.edu; borcea@njit.edu).

Digital Object Identifier 10.1109/TVT.2020.2983434

All recent solutions are based on accurate and real-time traffic information acquisition from smart phones or systems embedded in the cars [6], [7]. Typically, a central entity collects this information, achieves a global view of the traffic [4], and predicts better routes for drivers using current and historical traffic data [5]. The communication is done either over cellular networks or a combination of vehicular ad hoc networks (VANETs) and road-side units (RSUs).

Despite their benefits, the current solutions have a number of problems. First, the drivers' selfish choice for the optimal route in the current time slot shifts traffic congestion from one area to another [8]. Second, many solutions assume that the drivers will follow the recommended route, but this is not always the case. Third, there is a lack of solutions that combine analytic proofs of optimality with practicality in real-world scenarios. To solve these problems, new algorithms should be designed to jointly consider the drivers‘ behaviors on route selection, the road networks' load, and travel time. Furthermore, these algorithms must have solid theoretical underpinnings and should lend themselves to efficient implementations.

This article proposes an optimization model that minimizes the sum of the total travel time in the road network and a time representation of the traffic imbalance effects in the network. The constraints of this optimization are the traffic conditions and the drivers' behavior.

Unlike user-optimum traffic optimizations [10], which are expected to lead to best travel times for drivers, our solution leads to better fairness for drivers, since it achieves similar travel time for drivers with the same origin-destination (OD) pairs. Achieving fairness is important for widespread system adoption, as a system that is perceived as unfair will not be used by drivers. Also, the travel times for our solution are close to the optimal values. Furthermore, our solution balances the traffic and alleviates congestion, whereas typical user-optimum optimizations do not work well in case of congestion [21].

Unlike other system-wide traffic optimizations [14], our solution considers the effects of the driver behavior on traffic load (i.e., the drivers may not follow the suggested routes), and thus can work better in practice.

This paper presents an analytic formulation of the optimization problem, and an algorithm, Dynamic Interior Point Method (DIPM), that solves this optimization through driver rerouting.

DIPM benefits from a central server that computes driver routes, which is reachable via cellular networks or VANETs plus RSUs.

Specifically, the contributions of this article are:

- A system model that includes: (i) two types of communication architectures, car-to-cloud and VANET-to-cloud, to realize the implementation of our optimization; and (ii) Closed-form expressions of the road segments' traffic condition (congested or not) and the vehicles' route selection property (altruistic or selfish).
- A novel optimization model for vehicular traffic. The objective in the optimization is analytically modeled in a drift-plus-penalty framework, which considers not only the total travel time in the road network, but also the traffic load imbalance based on traffic conditions and drivers' behavior.
- A Dynamic Interior Point Method (DIPM) that solves the analytic formulation of our optimization model. This method is implemented as an iterative algorithm that computes rerouting alternatives for drivers; the rerouting helps achieve our system-wide optimization objective. We prove that the DIPM algorithm has a feasible solution, converges in a finite number of iterations, and has an acceptable computational complexity.
- Evaluations in MatLab and VanetMobiSim that demonstrate DIPM performance. Theoretical analysis and simulation results show that DIPM is fast and can work for real-time navigation systems. Theoretical analysis also suggest that DIPM can scale reasonably well. The results of extensive simulations with realistic urban maps and traffic scenarios demonstrate that DIPM outperforms other dynamic rerouting algorithms in terms of travel time. Furthermore, DIPM improves fairness when compared with a user-optimum approach [21].

The rest of the paper is organized as follows. Section II discusses related work on traffic congestion avoidance and traffic load balance. The system framework and the basic traffic models are described in Section III. Section IV presents the analytic formulation of our optimization model. The design and analysis of the DIPM algorithm are presented in Section V. Section VI contains the evaluation of DIPM. The paper concludes in Section VII.

II. Related Work

Unbalanced traffic, caused by the vehicles' choice of overlapping shortest routes, is one of the main reasons of traffic congestion. This type of traffic congestion can be reduced via path planning and navigation systems. Wang *et al.* proposed a highly practical vehicle rerouting system called Next Road Rerouting (NRR) to aid drivers in making the most appropriate next road choice toward avoiding unexpected congestion [9]. The rerouting process in NRR is based on a multi-agent 3-tier architecture, which includes a traffic operation center, traffic lights, and vehicles. Jeong *et al.* proposed a self-adaptive interactive navigation tool (SAINT), which was tailored for cloud-based vehicular traffic optimization on road networks [11]. In this system, the vehicles report their navigation experiences and travel paths to the vehicular cloud. Based on real-time road traffic conditions and vehicular trajectories, the vehicular cloud calculates the road segment congestion estimation. Although these navigation systems improve the classical path planning algorithms, they switch the congestion from one area to another, which postpones the occurrence of traffic congestion, rather than avoiding it. In addition, these works considered the assignment as a dynamic process; however, the route selection methods are seldom discussed.

Pan *et al.* presented five traffic rerouting strategies designed to be incorporated in a cost-effective and easily deployable vehicular traffic guidance system that reduces travel time [12]. The proposed strategies proactively compute individually-tailored rerouting guidance to be pushed to vehicles when signs of congestion are observed on their routes. An improved real-time path planning algorithm was proposed by Guo *et al.*, which dispatches the backlogged vehicles at weighted road intersections based on the back-pressure ratio [10]. These algorithms can reduce the traffic congestion on the road network to some extents. However, they lack an analytic model to describe the problem, and the navigation results are not proven to be optimal.

Some works have used an optimization model to study the global traffic balance. Cao *et al.* proposed a model to predict the probability of drivers‘ choice on routing results and then the online strategies automatically controlling the traffic lights' phases and duration to make sure that the vehicles have low traveling time [26]. Zhou *et al.* proposed a two-level hierarchical control framework for large-scale urban traffic networks [13]. For the application of this architecture in real world, model-based predictive control was utilized to obtain the best solutions. However, this work modified the traffic lights timing to achieve the optimal status, rather than controlling the traffic flow, an approach which may not be immediately applicable in practice.

The optimization model in [27] is presented for the minimization of the probability that vehicles arrive at their destinations after given deadlines and the minimization of the total travel time. Cao *et al.* proposed two optimization models to describe the problem [14]. The Probability Tail Model (PT model) aims to obtain an optimal path that minimizes the probability of arriving at a destination later than a predefined deadline. The Stochastic Shortest Path Problem With Delay Excess Penalty Model (SSPD model) had a deterministic travel fee and a random travel time which aimed to obtain an optimal path that minimizes the sum of these two types of cost, i.e., the total travel fee and the expected penalty for arriving at the destination later than the predefined deadline. This work finds the optimal solution based on the Partial Lagrange Multiple method. However, these models did not consider the case that individual drivers' selection can influence the future traffic status. On the basis of these two models, the authors proposed an intelligent routing algorithm to minimize the traffic jam occurrence by directing the paths of multiple vehicles cooperatively [15]. The traffic network optimum is achieved if the probability for spontaneous traffic jam occurrence over the entire road network during a given observation time period is minimized. This objective aims to minimize network breakdown probability, rather than the total travel time and the load imbalance as in our optimization.

TABLE I
SUMMARY OF THE MAIN MATHEMATICAL NOTATIONS

Notation	Description
$\mathbb{N}$	road network set
$\mathbf{R}$	road segment set
$\mathbf{I}$	road intersection set
r_i	road segment i
$\lambda_i(T)$	the inflow rate of vehicles coming from other segments to r_i in time slot T
$\mu_i(T)$	the outflow rate of vehicles leaving r_i for other segments in time slot T
ρ	time slot duration
$N_i(T)$	number of vehicles on road segment r_i in time slot T
L_i	length of road segment r_i
$v_i^{\text{leg}}(T)$	legal speed of road segment r_i in time slot T
$Q_i^{\text{in}}(T)$	the number of vehicles entering network from r_i in slot T
$Q_i^{\text{out}}(T)$	the number of vehicles exiting network on r_i in slot T
c_i	capacity of road segment r_i (number of vehicles)
$\alpha_i(T)$	traffic condition of road segment r_i in time slot T
$\beta_m^i(T)$	vehicle m's altruism for road segment r_i in slot T
$tte_m^i(T)$	vehicle m's travel time estimation on r_i in slot T
θ	threshold for ratio of altruistic to optimal travel time
s	average inter-vehicular distance

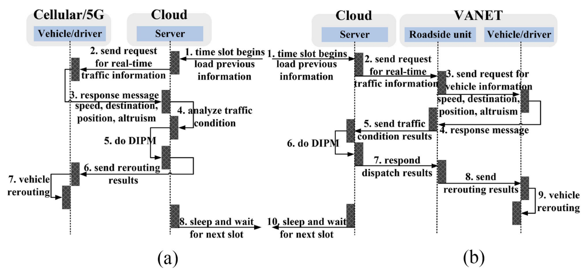

Cellular/5G
Vehicle/driver
Cloud
Server
1. time slot begins load previous information
2. send request for real-time traffic information
3. response message speed, destination, position, altruism
4. analyze traffic condition
5. do DIPM
6. send rerouting results
7. vehicle rerouting
8. sleep and wait for next slot
(a)
Cloud
Server
VANET
Roadside unit
Vehicle/driver
1. time slot begins load previous information
2. send request for real-time traffic information
3. send request for vehicle information speed, destination, position, altruism
4. response message
5. send traffic condition results
6. do DIPM
7. respond dispatch results
8. send rerouting results
9. vehicle rerouting
10. sleep and wait for next slot
(b)

Fig. 1. System operation under two architecture types. (a) Car-to-Coud Architecture. (b) VANET-to-Cloud Architecture.

Overall, our work differs from previous related work in two main aspects: (1) we propose an analytic optimization model that considers the traffic conditions and drivers' behavior on rerouting decisions, and (2) we present an algorithm that solves this optimization. The algorithm periodically optimizes the traffic flow for the road network, and thus minimizes the sum of the total travel time and a time representation of load imbalance effects for all drivers in the road network.

III. System Model

This section presents an overview of our system operation and formulates the models of time-varying traffic flow and congestion estimation on road segments. The notation used in this section is described in Table I.

A. Transportation Architecture and System Operation

As shown in Fig. 1, our system can work with either: (a) direct communication between the cars and the server using cellular/5 G communication or (b) ad hoc communication (VANET to RSU) and cellular/wired communication (RSU to server), which is based on our previous work [10]. The first architecture is simpler, whereas the second one is more scalable.

In both cases, the system periodically (i.e., in each time slot) collects real-time traffic information, analyzes the traffic conditions and generates the input parameters for DIPM, and

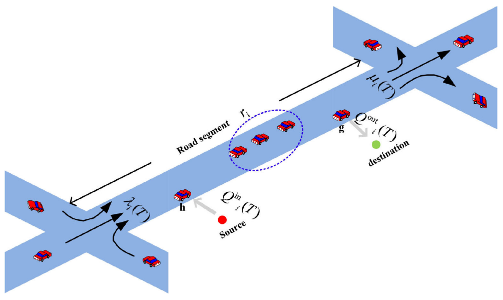


Fig. 2. Illustration of traffic flow on road segment.

then sends the rerouting results generated by DIPM to vehicles. The RSUs in the VANET-to-Cloud architecture aggregate the results during the traffic collection phase and disseminate the rerouting results to cars in their coverage area. In this way, the load on the server can be reduced at the the expense of longer delays and potentially lost messages in VANET.

In our system, the road network is defined as a weighted graph. We assume that each road segment between two road intersections is unidirectional; a bidirectional road is split into two unidirectional segments. For brevity, we use "road segments" to refer to "unidirectional road segments" in the rest of the paper. With assistance from RSU, the number of vehicles on each road segment can be estimated within an acceptable error for different traffic densities, even when some vehicles are not part of the system.

The behavior of the drivers is modeled using the binary "altruism" parameter, which tells DIPM if a driver is expected to follow the rerouting suggestion or not. To reduce load imbalance and implicitly traffic congestion, the suggested route for some drivers may be slightly longer than the optimal route. If they take this route, we consider them altruistic. If they take the optimal route (using their own knowledge), we consider them selfish. This "altruism" parameter is learned over time by analyzing the driver's decisions to rerouting suggestions.

B. Traffic Flow Model

We model the vehicles' traffic flow as an inflow/outflow system with sequential time slots. As shown in Fig. 2, the number of vehicles on road segment r_i in the current time slot can be divided into three components: the backlogged vehicles from the previous time slot, the inflow and outflow of vehicles from/to the neighbor road segments, and the vehicles that enter the road network from this segment or have a destination on this segment (i.e., exit the road network on this segment).

Parameters $\lambda_i(T)$ and $\mu_i(T)$ denote the inflow rate and outflow rate, respectively, of road segment r_i in time slot T. They have variable values in different time slots; however, they are regarded as constant in each time slot. The parameter ρ is the duration of each time slot. The traffic volume that starts and departs from r_i in time slot T is $\rho[\lambda_i(T) - \mu_i(T)]$. The lower bound of ρ should be larger than the computing time for one DIPM calculation to make sure the vehicles in the road network can use the results. Furthermore, ρ should not be too low to avoid unnecessary calculations and waste of system resources in relatively stable networks. On the other hand, if ρ is too large, the system cannot set up the optimization model based on the latest traffic parameters due to the changes in the real-time traffic conditions, which will influence the performance of traffic load balance. Therefore, our system will set ρ as a multiple of the traffic light's period, based on the number and speed of the vehicles in the road network.

The number of vehicles that enter and exit the road network on r_i in time slot T are defined as: $Q_i^{\text{in}}(T)$ and $Q_i^{\text{out}}(T)$. Vehicle h and vehicle g illustrate this type of traffic in Fig. 2. The number of backlogged vehicles from the previous time slot is $N_i(T-1)$, illustrated by the vehicles in the circle in the figure. Specifically, $N_i(T-1)$ denotes the number of starting vehicles inside road segment r_i at the beginning of the time slot.

Therefore, the number of vehicles on road segment r_i in time slot T is:

$$\begin{aligned} N_i(T) = N_i(T-1) + \rho[\lambda_i(T) - \mu_i(T)] \\ + [Q_i^{\text{in}}(T) - Q_i^{\text{out}}(T)]. \end{aligned} \tag{1}$$

Let us note the road segment's total inflow in each time slot is limited by c_i, the capacity of road segment r_i:

$$\rho\lambda_i(T) + Q_i^{\text{in}}(T) \le c_i. \tag{2}$$

Here, the capacity c_i of a road segment can be acquired from its associated RSU's cache, if RSUs are deployed, or calculated based on the segment length and the average inter-vehicular distance as $c_i = \lceil \frac{L_i}{d_{\text{inter}}} \rceil$.

C. Congestion Estimation

The traffic condition is determined by the relation among density (the number of vehicles per distance unit), flow (the number of vehicles per time unit), and mean speed [22].

Congestion definition 1 (speed-volume): When the speed on road segment r_i in time slot T is below a given threshold value and the slope of flow is negative, the road segment r_i in time slot T is congested. The formal definition is:

$$v_i(T) \le v_i^{\text{thr}}(T), \quad \frac{\partial q}{\partial k} \le 0 \tag{3}$$

The drawback of this definition is that the threshold value is difficult to define. Commonly, this value is set to two-thirds of the legal maximum speed on the road segment. However, since this value is arbitrary, we consider the following congestion definition for the rest of the article.

Congestion definition 2 (number of backlogged vehicles): The road segment r_i is congested in time slot T when the number of backlogged vehicles in the previous time slot is larger than the sum of outflow and exit vehicles in the current time slot. The definition can be described as:

$$N_i(T-1) > \rho\mu_i(T) + Q_i^{\text{out}}(T). \tag{4}$$

Using this definition, the traffic condition of a road segment in a given time slot, $\alpha_i(T)$, is defined as:

$$\begin{aligned} \alpha_i(T) = \varepsilon[N_i(T-1) - \rho\mu_i(T) - Q_i^{\text{out}}(T)] \\ - \varepsilon[\rho\mu_i(T) + Q_i^{\text{out}}(T) - N_i(T-1)] \end{aligned} \tag{5}$$

Here, $\varepsilon(\cdot)$ is a unit step function. When the road segment is congested in time slot T, the value of $\alpha_i(T)$ is 1. Otherwise, the value of $\alpha_i(T)$ is -1.

IV. Analytic Formulation of Optimization Model

When all drivers in the road network choose their own optimal route as perceived at a given time, traffic congestion may occur because the drivers do not consider the future traffic load on the roads. While user-optimum optimizations that consider future traffic load exist [10], they do not work well in practice for three reasons: (i) they are unfair, as drivers with the same OD pairs may end up with very different travel times, and this may cause problems with system adoption, (ii) they are computationally expensive and the results may arrive too late for effective driver rerouting, and (iii) they do not work well during congestion, which sometimes is unavoidable [21].

Therefore, this article focuses on a system-wide optimum solution to the traffic congestion problem. Existing system-wide travel time optimizations suffer from two problems. They either do not consider the driver behavior when making decisions [14], or do not provide proofs of optimality [12]. Our solution addresses both problems.

The rest of this section presents a discussion of drivers' behavior and alternative route constraints, followed by the analytic formulation of our optimization, which aims to minimize the sum of the total travel time in the road network and a time representation of the traffic imbalance effects in the road network.

A. Driver Behavior and Alternative Route Constraints

Given our optimization goal, it is expected that most drivers will end up with optimal routes, but a small number of drivers will not be recommended optimal routes. In general, drivers do not know whether the route recommended by our system is optimal or not. However, they know that the system optimizes the total travel time and load balance, and that in general they will end up with a shorter travel time when they follow the system's recommendation. Nevertheless, we assume that some drivers will not follow the route suggested by the system and will, instead, follow their individual optimal route.

To model this behavior, we use a parameter $\beta_m^i(T)$. The value of $\beta_m^i(T)$ is 1 when the driver of vehicle m chooses its optimal road segment r_i in time slot T, which means the driver is selfish for the road network. The value of $\beta_m^i(T)$ is -1 when the driver of vehicle m chooses the alternative road segment r_i in time slot T, where r_i is a non-optimal segment suggested by our system; in this case, we say the driver is altruistic for the road network. The driver altruism can be further defined as:

$$\begin{aligned} \beta_m^i(T) = 2\varepsilon[tte_m^{\text{opt}}(T) - tte_m^i(T)] \\ - \varepsilon[tte_m^i(T) - tte_m^{\text{opt}}(T)]. \end{aligned} \tag{6}$$

Here, $tte_m^{\text{opt}}(T)$ denotes the vehicle m's optimal travel time in time slot T, and $tte_m^i(T)$ is the travel time when vehicle m chooses the non-optimal road segment r_i in time slot T, with $tte_m^i(T) \ge tte_m^{\text{opt}}(T)$. $\varepsilon(\cdot)$ is a unit step function, which returns 0 when the variable is negative, and 1 otherwise. The vehicles that are not in our system are considered to have selfish driver behavior with $\beta_m^i(T) = 1$. Given our global optimization goal, the number of vehicles selecting non-optimal routes will be smaller than the number of vehicles selecting optimal routes:

$$\sum_{m=1}^{q} \beta_m^i(T) \geq 0. \tag{7}$$

Here, q is the number of vehicles. If the travel time of the altruistic road segment is significantly higher than the optimal segment's, the drivers may decide to stop using our system. Therefore, the travel time of the altruistic road segment will have an upper bound, described as:

$$E_m^i(T) \leq \theta tte_m^{\text{opt}}(T). \tag{8}$$

Here, $E_m^i(T)$ is the extra travel time of vehicle m, which chooses the altruistic road segment r_i in time slot T. It will not exceed θ percent of the optimal segment's travel time of vehicle m. The parameter θ relates to the drivers' acceptance. In our paper, its default value equals 0.3 (i.e., 30%).

The detailed method to acquire the travel time estimation of both tte_m^{opt} and tte_m^i is based on our previous work [10]. Our rerouting algorithm is improved based on Yen's algorithm [28], which considers the futile rerouting issue. If the potential rerouting path has a sub-path that is part of the previous shortest path, the algorithm will remove the edges of this sub-path.

B. Global Traffic Optimization

The objective in our optimization is analytically modeled in a drift-plus-penalty framework, which considers not only the total travel time in the road network, but also the traffic load imbalance based on traffic condition and drivers' behavior.

The main objective of the optimization is to reduce the total travel time in the road network. The travel time of vehicle m on road segment r_i in time slot T is defined as followed, based on work done in [16]:

$$tte_m^i(T) = \frac{L_i}{v_i^{\text{leg}}(T)} \left[1 + k_1 \left(\frac{N_i(T)}{c_i}\right)^{k_2}\right]. \tag{9}$$

Here, L_i denotes the length of road segment r_i. $v_i^{\text{leg}}(T)$ is the legal speed of road segment r_i, which is the free flow speed. k_1 and k_2 are adjustment parameters, which define the relation between the speed and the traffic density quantitatively. The values of k_1 and k_2 depend on the traffic scenario, which can be estimated via evaluation and are not the same for all road networks (we list the values used in our evaluation in Section VI). Therefore, the total travel time of all the vehicles in the road network is:

$$\sum_{i=1}^{p} \sum_{m=1}^{q} tte_m^i(T). \tag{10}$$

Here, p denotes the number of road segment and q denotes the number of vehicles. To avoid low-accuracy estimates of TTE when the number of vehicles on the road segment is inaccurate, we leverage the work in [10], which adjusts the results of TTE for different system penetration.

Our secondary optimization objective is to reduce the traffic load imbalance. The drivers' behavior for the route selection (altruistic or selfish) and the traffic conditions will both influence the traffic load imbalance in the road network. Considering these two parameters, there are four cases that must be analyzed:

- *Case 1:* The road segment r_i is not congested in time slot T, and vehicle m selects r_i as its altruistic road segment.
- *Case 2:* The road segment r_i is not congested in time slot T, and vehicle m selects r_i as its optimal road segment.
- *Case 3:* The road segment r_i is congested in time slot T, and vehicle m selects r_i as its optimal road segment.
- *Case 4:* The road segment r_i is congested in time slot T, and vehicle m selects r_i as its altruistic road segment.

Next, we present analytic formulations to analyze the impacts of these four cases. The parameter $\alpha_i(T)$ describes the traffic condition of road segment r_i in time slot T. As shown previously, the vehicle m's selection (altruistic/selfish) is described by $\beta_m^i(T)$. The impact of a vehicle's route selection can be estimated by the existed parameters:

$$E_m^i(T) = tte_m^i(T) - tte_m^{\text{opt}}(T). \tag{11}$$

Furthermore, the impact of the road segment's traffic condition on the global traffic load can be estimated as:

$$B_m^i(T) = \max\left\{0, \frac{L_i}{v_i^{\text{leg}}(T)} \left[1 + k_3 \left(\frac{N_i(T) - N_i^{\text{thr}}(T)}{c_i}\right)^{k_4}\right]\right\}, \tag{12}$$

which means that if the road segment r_i is not congested in time slot T, the number of vehicles on the road segment r_i is below the threshold value, and the vehicles on this segment have no impact on the global traffic load in this time slot. Otherwise, $B_m^i(T)$ is the travel time increase for the road segment r_i. The congestion threshold we use is based on what we estimate in Eq. (4). In addition, the definitions guarantee that $B_m^i(T)$ and $E_m^i(T)$ have the same unit of measurement (second), in order to be summed up with the travel time in our optimization. The values of k_3 and k_4 depend on how much traffic load balance the system wants to achieve. If the system wants to balance the traffic load more, the values of k_3 and k_4 will be larger. In this paper, we define $k_1 = k_3$, $k_2 = k_4$.

By combining the parameters $\alpha_i(T)$ and $\beta_m^i(T)$, the impact of the four cases can be estimated quantitatively:

$$\sum_{i=1}^{p} \sum_{m=1}^{q} \left(\alpha_i(T) B_m^i(T) + \beta_m^i(T) E_m^i(T)\right). \tag{13}$$

Where p is the number of road segments, and q is the number of vehicles in the road network.

At this stage, we can model our multi-objective global traffic optimization as follows.

Given:

1) Road network $\mathbb{N} = \{\textbf{RS}, \textbf{RI}\}$
2) The real-time traffic information of each road segment in time slot T, including the inflow/outflow rate, the number

of enter/exit vehicles, and the backlogged vehicles from previous time slot.

3) The vehicles' real-time positions and their destinations.

Objective:

$$\min \sum_{i=1}^{p}\sum_{m=1}^{q}\left[tte_m^i(T)+\alpha_i(T)B_m^i(T)+\beta_m^i(T)E_m^i(T)\right] \tag{14}$$

Subject to: (1), (2), (7), and (8).

This objective is to minimize the sum of the total travel time and a time representation of the effects of load imbalance. The traffic load is impacted by drivers 'behavior and road segments' traffic conditions as shown in Eq. (13). The constraints for our proposed objectives are: 1) road segments' traffic flow conservation between two sequential time slots; 2) upper bound of the number of inflow vehicles; 3) upper bound of the altruistic vehicle ratio; 4) upper bound of alternative route's extra travel time. The proposed model presents the dynamic relationship between time slot $T-1$ and time slot T, until all the vehicles in the road network reach their destinations.

Solution: The model outputs the optimal next road segment and the suggested/altruistic next road segment for each vehicle. Oftentimes, the suggested segment is the optimal one. In the other cases, the suggested segment is longer than the optimal one. In this case, the β_m^i parameter models the driver behavior in time slot T; it tells us whether the driver chooses the recommended altruistic road segment or the optimal road segment.

V. DIPM Design and Analysis

This section presents our Dynamic Interior Point Method to solve the optimization model presented in Section IV, as well as its performance analysis.

A. Algorithm Design

Our optimization model contains three inequality constraints and one equality constraint. This model is of the same type as the one in [17]:

$$\begin{aligned}&\text{minimize } f_0(x)\\&\quad\text{s.t.}\\&\qquad Ax=b\\&\qquad f_a(x)\le 0,\ a=1,2,\ldots n.\end{aligned} \tag{15}$$

In all the functions of our model, $N_i(T)$ can be regarded as x in Eq. (15). $f_0(x)$ is the objective function in Eq. (14), which denotes $tte_m^i(T)+\alpha_i(T)B_m^i(T)+\beta_m^i(T)E_m^i(T)$. The n in Eq. (15) is the number of inequality constraints, which is 3 in our case. Therefore, $f_1(x)$ is our Eq. (2), $f_2(x)$ is our Eq. (7), and $f_3(x)$ is our Eq. (8). The equality constraint $Ax=b$ is our Eq. (1).

This type of formulation for our optimization model can be analytically solved by the Interior Point Method (IPM) [17]. However, as we will explain later, using this method directly is not sufficient for our problem because it can be applied for only one time slot, and thus cannot guarantee that all the vehicles arrive at their destinations when the method finishes. Furthermore, we need to design an algorithm for a computational solution to our problem. Therefore, we propose the Dynamic Interior Point Method (DIPM), which is an algorithm that works for road networks and all time slots, until all vehicles reach their destinations. In the following, we explain how we use the original IPM and how we incorporate it in the DIPM algorithm.

IPM uses the indicator function to offset the inequality constraints, which means that the inequations in constraints $f_a(x)\le 0$ are transformed into indicator functions $I_-(f_a(x))$. The indicator functions are represented as:

$$I_-(u)=\begin{cases}0, & u\le 0\\ \infty, & u>0.\end{cases} \tag{16}$$

Then, the model in Eq. (15) can be reformulated as an equality constrained problem:

$$\begin{aligned}&\min\left(f_0(x)+\sum_{a=1}^{3}I_-(f_a(x))\right)\\&\text{s.t.}\\&\qquad Ax=b\end{aligned} \tag{17}$$

This indicator function $I_-(f_a(x))$ can be approximated via a logarithmic barrier, which contains a parameter t:

$$\sum_{a=1}^{3}I_-(f_a(x))\approx-\frac{1}{t}\sum_{a=1}^{3}\log(-f_a(x)) \tag{18}$$

Then, the Eq. (17) can be approximated as:

$$\begin{aligned}&\min\quad\left(f_0(x)-\frac{1}{t}\sum_{a=1}^{3}\log(-f_a(x))\right)\\&\text{s.t.}\quad Ax=b\end{aligned} \tag{19}$$

Therefore, the inequality constrained problem in Eq. (15) is transformed into Eq. (17), and then approximated into Eq. (19). The Eq. (19) is an equality constrained problem, which can be solved with the Lagrange function under the KKT conditions [18].

The Lagrange function that considers $N_i(T)$ as variable is:

$$\begin{aligned}&\mathscr{L}[N_i(T),\zeta^*(t),\nu^*(t)]\\&=f_0(N_i(T))+\sum_{a=1}^{3}\zeta_a^*(t)f_a(N_i(T))+\nu^*(t)(AN_i(T)-b).\end{aligned} \tag{20}$$

Here, $\zeta^*(t)$ and $\nu^*(t)$ denote the dual operators of the Lagrange function; $\zeta^*(t)$ is $-\frac{1}{tf_a^*(N_i(T))}$ and $\nu^*(t)$ is $\frac{w}{t}$. A and b denote the coefficients of the constraint in equation (1).

Because IPM cannot take into consideration the traffic flow across consecutive time slots, it can only be used for each individual time slot, using the real-time data as static variables in KKT conditions. Though it is possible to apply IPM repeatedly at the beginning of every time slot, as implemented in Section VI as a competing mechanism of DIPM, this method still cannot detect the traffic flow relationship between time slots, and thus cannot stop the repeated calculation automatically when all the vehicles arrive at their destinations.

DIPM improves the coefficient A as a dynamic variable that relates to the current time slot T and the previous time slot $T-1$ in the barrier function:

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta N_i(T) \\ \Delta w \end{bmatrix} = - \begin{bmatrix} \nabla f(N_i(T)) \\ A(N_i(T)) - b \end{bmatrix}, \tag{21}$$

where H can be described as

$$\begin{aligned} H &= \text{diag}[t\nabla^2 f_0(N_i(T)) - \sum_{a=1}^{3} \nabla^2 f_a(N_i(T))] \\ &= \text{diag}\left[[t\gamma_1 + (1-\gamma_3(1+\theta))] \frac{L_i k_1 k_2 (k_2 - 1)}{(c_i)^2 v_i^{leg}(T)} \right. \\ &\quad \left. - \frac{\beta_i^m(T) L_i k_3 k_4 (k_4 - 1)}{(c_i)^2 v_i^{leg}(T)} \left[\frac{N_i(T) - N_i^{thr}(T)}{c_i} \right]^{k_4} \right]. \end{aligned} \tag{22}$$

The IPM algorithm considers A as a unit matrix based on equation 1, and the b coefficient as $N_i(T-1) + \rho[\lambda_i(T) - \mu_i(T)] + [Q_i^{in}(T) - Q_i^{out}(T)]$. Since IPM works only in an individual time slot, it loads $N_i(T-1)$ as a part of variable b.

Unlike IPM, DIPM works for road networks and all time slots, and it considers the dynamic flow in current time slot T and the previous time slot $T-1$. The coefficient A is set as a variable that relates dynamically to the time slot via $\Delta N_i(T)$. In DIPM, the variable b in each time slot is $\rho[\lambda_i(T) - \mu_i(T)] + [Q_i^{in}(T) - Q_i^{out}(T)]$, which transforms $N_i(T-1)$ as $\Delta N_i(T)$ via $\Delta N_i(T) = N_i(T) - N_i(T-1)$. A is $I - \gamma(T)$ when $N_i(T-1) = \gamma(T) N_i(T)$. Therefore, A is a variable that relates to the time slot T and $\Delta N_i(T)$ in the barrier function. The equation (21) in DIPM is transformed as:

$$H = \text{diag}\left[t\nabla^2 f_0(N_i(T)) - \sum_{a=1}^{3} \nabla^2 f_a(N_i(T)) \right] \tag{23a}$$

$$A = I - \gamma(T) \tag{23b}$$

$$b = \rho[\lambda_i(T) - \mu_i(T)] + [Q_i^{in}(T) - Q_i^{out}(T)] \tag{23c}$$

Here, I is a unit matrix. $\gamma(T)$ is the ratio matrix between $N_i(T-1)$ and $N_i(T)$ for $i \in \{1, 2, \dots m\}$.

In addition to improving the definitions of variables A and b to make them work for all the time slots and the whole road network, DIPM improves the stop criteria used by IPM. The IPM algorithm has two stop criteria:

- The error between the original optimal results and the IPM's results is smaller than the threshold value. When the parameter t increases during the iteration, the error between the approximated optimization model in Eq. (19) and the original model in Eq. (15) will be reduced iteratively. Therefore, the algorithm will stop at the iteration when the error is under the threshold value. The default value for the error threshold is 10^{-6}.
- The number of iterations reaches a maximum value. In our case, the algorithm's processing time cannot be larger than the duration of the time slot because we need to make timely rerouting decisions. Thus, IPM needs a constraint on the number of iterations. The default value for the number of iterations is 1000.

Algorithm 1: DIPM Pseudo Code (Per Time Slot).

```
1:  Input the road network $\mathbb{N} = \{\mathbf{RS}, \mathbf{RI}\}$ as matrix
2:  Input error threshold $\epsilon > 0$, maximum iteration
    number $j_{\max}$
3:  Load the real-time traffic status of road network
4:  Initialize road segment id $i = 1$, iteration time $j = 1$
5:  Transform the variables $N_i(T)$ and $\alpha_i(T)$
6:  Input vehicles' OD pairs
7:  /*Create the optimization model*/
8:  Create the objectives as Eq. (14)
9:  Set up four constraints Eq. (1) Eq. (2) Eq. (8) Eq. (7)
10: /*Search the optimal arrangement result via IPM*/
11: Transform Eq. (2), Eq. (8), and Eq. (7) via indicator
    function as Eq. (17).
12: Approximate indicator function as Eq. (19)
13: Set up the Lagrange multiplier and dual multiplier
    $\zeta^*(t)$, $\nu^*(t)$ via Eq. (2), Eq. (8), and Eq. (7)
14: Create Lagrange function $\mathscr{L}[x, \zeta^*(t), \nu^*(t)]$.
15: /*Do iterations until any stop criteria is satisfied*/
16: while $\frac{3}{t} \geq \epsilon \&\& j \leq j_{\max}$ do
17:   /*$\frac{3}{t} \geq \epsilon$: error is larger than the threshold*/
18:   /*$j \leq j_{\max}$: maximum number of iterations not
      reached*/
19:   if $N_i = 0$ for all $i \in \{1, 2, \dots m\}$ then
20:     /*All vehicles in road network reach destinations*/
21:     Go to line 30
22:   else
23:     Find feasible solution of $\mathscr{L}[x, \zeta^*(t), \nu^*(t)]$
24:     Calculate $H$, $A$, $b$, $N_i(T)$ as $x$ via eq. (23)
25:     $x = x^*(t)$
26:     $t = jt$
27:     $j++$
28:   end if
29: end while
30: Output the optimal rerouting results for each driver
```

DIPM adds one more stop criterion for termination: the execution ends when all the vehicles in road network reach their destinations. This criterion makes sense in experimental evaluations; in real-life, the algorithm will run continuously.

IPM can calculate the optimal results in one time slot. DIPM, on the other hand, adapts to the traffic dynamically, as it takes the results from the previous time slot and uses them in the current time slot. The full comparison between DIPM and IPM, which illustrates the improvements of DIPM is presented in Table II.

Algorithm 1 presents the pseudo code of DIPM in one time slot. At the beginning of the time slot, Lines 1$\sim$6 initialize the model's input and collect real-time traffic information. The algorithm sets up the optimization model via lines 8$\sim$15. Lines 16$\sim$24 execute iterations to obtain the optimal results for DIPM, and then the algorithm ends by outputting the rerouting results for all drivers.

B. Algorithm Analysis

Theorem 1: Our optimization model has a feasible solution and the optimal result is attained.

TABLE II
COMPARISON OF IPM AND DIPM

Comparison criteria	IPM	DIPM
A	load data via eq.(1)	set up A with $\Delta N_i(T)$
b	$N_i(T-1)+\rho[\lambda_i(T)-\mu_i(T)]+[Q_i^{in}(T)-Q_i^{out}(T)]$	$\rho[\lambda_i(T)-\mu_i(T)]+[Q_i^{in}(T)-Q_i^{out}(T)]$
Step length	constant	time-vary variable among time slots
Stop criterion (a)	iteration threshold	iteration threshold
Stop criterion (b)	error threshold	error threshold
Stop criterion (c)	-	no vehicle left in the system
Can it work for sequential time slots?	No	Yes

Proof: See Appendix A. ■

Theorem 2: DIPM converges to optimal results within a finite number of iterations.

Proof: See Appendix B. ■

Theorem 3: The computational complexity of DIPM is $O(KN(p+N\log N)+s+\sqrt{n}\log(\frac{n}{\epsilon}))$, where K is the number of routes provided to vehicles; K is set to 2 in our experiments (optimal route and alternative route), but can be higher if we want more alternative routes. p is the number of road segments, N is the number of road intersections. s is cardinality of the set of OD pairs for all drivers, where the origin for each driver is considered its current segment. n is the number of inequational constraints and ϵ is the error threshold.

Proof: See Appendix C. ■

Remark 1: Compared to the method that repeatedly applies IPM over all time slots, DIPM reduces the computational complexity by $O(\Delta z(\sqrt{n}\log(\frac{n}{\epsilon})))$. Δz denotes the reduction of the number of time slots. In each time slot, DIPM calculates one less parameter (i.e., A).

Proof: See Appendix C. ■

These theorems and the remark demonstrate that DIPM can effectively and efficiently find the optimal number of vehicles on each road segment in each time slot in order to reduce both the total travel time and the traffic load imbalance in the network as much as possible.

VI. EVALUATION

We evaluate DIPM using simulations. The evaluation aims to answer the following three questions: 1) Is DIPM fast enough to work in a real-time navigation system? 2) Does DIPM performs better than existing practical solutions, including IPM, dynamic LET, etc, in terms of travel time? and 3) Does DIPM improve driver fairness when compared to existing practical solutions and a user-optimum traffic optimization?

A. Experiment Setup

We conduct our evaluation using MATLAB and VanetMobiSim [24], which is a traffic pattern and vehicle mobility simulator. Specifically, we implement and run DIPM and the following competing algorithms in MATLAB:

- Static shortest path (SSP) [10]: A static path planning algorithm that chooses the shortest path based on the traffic information when the vehicle enters the road network; no rerouting happens during the travel.
- Entropy-balanced k shortest path (EBkSP) [12]: An improved kSP algorithm that takes into consideration the impact of each path selection on the density of the affected road segments.

TABLE III
SYNTHETIC ROAD NETWORKS PARAMETERS

road segments	road intersections	2-connected segments	3-connected segments	4-connected segments
100	31	4	16	11
200	57	3	22	32
300	84	7	22	55
400	110	4	31	75
500	135	4	32	99

- A* shortest path with repulsion (AR*) [12]: An enhanced A* algorithm with a weighted vehicle footprint counter and an improved heuristic function.
- Dynamic traffic assginment (DTA) [21]: A dynamic traffic assignment method that achieves user equilibrium. Despite not being a viable solution for real-time traffic guidance [12], DTA is valuable because it gives us a lower bound on the travel time and allows us to compare DIPM's fairness against a well-known user-optimum approach to traffic optimization.

The vehicular mobility model is set as polito.uomm extension in VanetMobiSim. The number of vehicles entering the road network in different time slots is generated using a Normal distribution. The system sets an initial number of vehicles at the beginning of simulation, which considers vehicles starting inside the road network. The execution of an algorithm in MATLAB generates rerouting results, which are fed into VanetMobiSim. Based on the rerouting results, VanetMobiSim simulates the movement of the vehicles on a map and generates a set of vehicular trace files as the result of simulation. We measure the execution time of DIPM in MATLAB, and analyze the trace files generated by VanetMobiSim to evaluate the DIMP's solutions quality. Both MATLAB and VanetMobiSim run on a Windows10 computer with Core i7 processor and 8 Gb memory.

The execution time of DIPM is mainly determined by the complexity of road network. Therefore, for experiments regarding execution time, we use MATLAB to generate synthetic road networks with different numbers of road segments and intersections, as summarized in Table III.

To measure the travel time and fairness, we use a real map from OpenStreetMap [23]. The map (shown in Fig. 3) is a part of the road network in Newark, NJ, USA. The details of the map are summarized in Table IV. This table also contains a few other key parameters needed by VanetMobiSim for generating traffic and conducting simulations. The traffic generation model is the same with the one in [10].

B. DIPM Execution Time

Our goal with these experiments is two-fold: (i) verify if the measured execution time of DIPM matches the theoretically

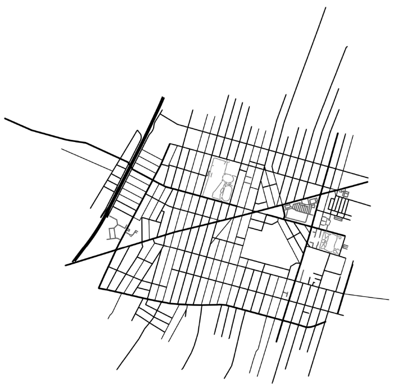


Fig. 3. Map used in simulations: area of Newark, NJ, USA.

TABLE IV
SIMULATION PARAMETERS FOR REALISTIC SCENARIO

Simulation parameter	Value
network area (km^2)	27.09
number of road segments	1494
number of road intersections	505
total segment length (km)	201.83
number of vehicles	1000
origin-destination (OD) pairs	100
traffic light duration (s)	100
time slot duration (s)	100
number of traffic lights	85
vehicular legal speed (km/h)	40 / 50 / 70 / 80 / 100 / 130
travel recording interval (s)	0.1
simulation time in VanetMobiSim (s)	20000
$k_1/k_2/k_3/k_4$	0.35 / 0.6 / 0.35 / 0.6

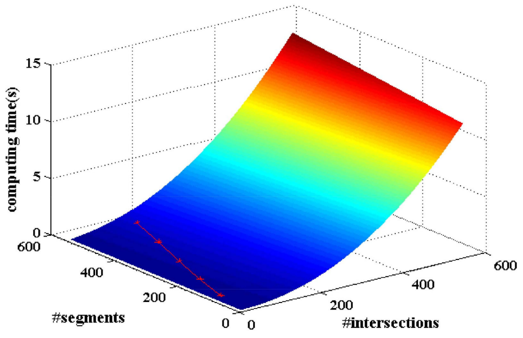


Fig. 4. DIPM execution time: the plane is the theoretically estimated time, and the curve is the measured time.

-estimated execution time, and (ii) understand if the execution time of DIPM is good enough for a real-time navigation system.

Fig. 4 shows a 3D visualization of the execution time of DIPM for road networks of different sizes. The plane shows the execution time estimated theoretically based on the analysis of the algorithm, for all combinations of the number of road segments (100–600) and the number of intersections (100–600). The curve shows the actual execution time measured on our platform for 5 map configurations (100 segments-31 intersections, 200 segments-57 intersections, 300 segments-84 intersections 400 segments-110 intersections, and 500 segments-135 intersections); each data point on the curve is the average execution time of 20 runs of DIPM with 5 different sets of OD pairs and different initial values.

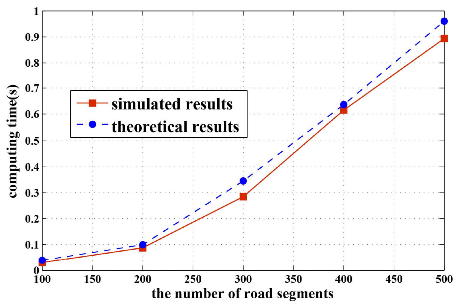


Fig. 5. DIPM execution time: estimated time vs. measured time for 5 map configurations.

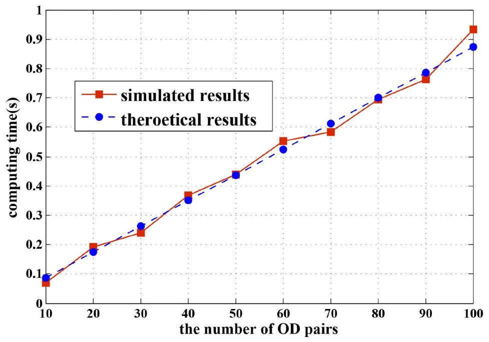


Fig. 6. DIPM execution time: estimated time vs. measured time for different numbers of OD pairs.

The results in Fig. 4 show that execution time measured in the experiments is very close to the theoretically-estimated execution time. Fig. 5 shows a 2D visualization of this result. The estimated time is plotted for the same map configurations as the measured time. The results illustrate that the difference between the theoretically-estimated execution time and the measured execution time is less than 5%.

Fig. 6 shows how the execution time changes with the number of OD pairs. The map used is the one with 500 road segments in Table III; the number of vehicles is 1000, and their origin and destination locations are randomly selected from the OD pairs based on normal distribution. The results demonstrate that the two types of execution time match very well. Furthermore, the execution time increases roughly linearly with the number of OD pairs.

Thus, even though it is not realistic to generate maps of all sizes and repeat the experiments with these maps, we can use the estimated execution time from the theoretical analysis to quantify how DIPM scales. First, let us note that DIPM's execution time for a fairly large map with 600 segments and 600 intersections is 15 seconds on a laptop. Since the travel time within a road segment usually exceeds 15 seconds, we conclude that DIPM is fast enough to generate solutions in real-time for such networks. Based on the DIPM's theoretical analysis, we also estimate that DIPM can work for much larger networks, but it will require more powerful hardware, which is readily available (e.g., in the cloud).

Fig. 7 demonstrates that DIPM works better than just invoking IPM repeatedly over the time sots. Specifically, the figure shows

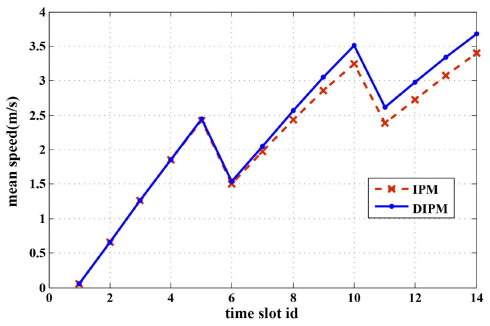


Fig. 7. Average speed comparison between DIPM and IPM in different time slots.

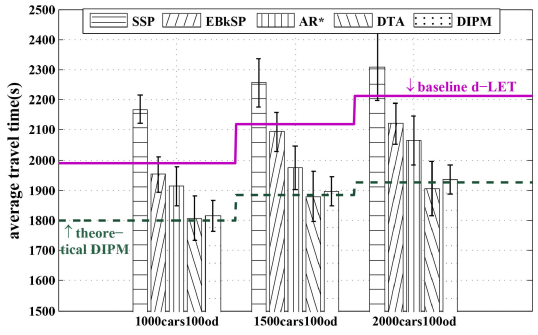


Fig. 8. Comparison of average travel time for 3 traffic scenarios.

a comparison of the average speed for all vehicles in different time slots. We observe that DIPM results in higher speed, and the difference between DIPM's speed and IPM's speed keeps increasing over time. The average speeds of IPM and DIPM at the beginning are the same. Because the number of vehicles is small at the beginning, $N_i(T-1)$ equals zero, which means that the variables A and b in both DIPM and IPM are the same. Thus, the difference between the two algorithms is negligible at the beginning. However, after several time slots, the advantage of DIPM becomes apparent because it can provide optimal dispatch results dynamically as the number of time slots grows. This is explained by the dynamic setting of variables A and b in DIPM. Furthermore, DIPM terminates within 14 time slots. Under IPM, with these time slots, 12.23% of vehicles cannot arrive at their destinations; for these vehicles to arrive their destinations eventually, as many as 18 time slots are needed.

C. Travel Time Comparison

Fig. 8 shows the comparison of the average travel time (per driver) between DIPM and 4 comparison algorithms, using 3 traffic scenarios. We repeat each simulation 5 times. Across all scenarios, DIPM performs better than the 3 practical solutions, namely AR*, EBkSP, and SSP. Compared to these algorithms, the average travel times with DIPM are 19.2%, 9.15%, and 5.39% shorter, respectively. While the average travel time increases with the number of vehicles (i.e., the traffic slows down), the performance advantage of DIPM remains stable. For example, compared to SSP, the average travel time with DIPM is shorter by 19.41%, 18.9% and 19.29% for 1000 vehicles, 1500 vehicles, and 2000 vehicles, respectively.

Furthermore, based on [25], we added the performance of "Dynamic LET" in Fig. 8 as an intuitive baseline, which is shown as a line in the figure. "Dynamic LET" performs path planning based on Least Expected Time in each time slot in a greedy way. The simulation results show that the SSP's average traveling time bar is above the baseline because SSP does not provide rerouting in the following time slot. The AR* and EBkSP are below the baseline because the rerouting of AR* and EBkSP considers the influence of vehicles' path choices on the future traffic conditions. As explained above, DIPM performs best and it is substantially better than Dynamic LET. Furthermore, the line named theoretical DIPM shows the theoretical objective values. The difference between the theoretically-calculated average travel time and the simulated average travel time is less than 1%. The error is caused by two factors: the error of parameter tte estimation, and a minor delay at vehicles to receive the optimal results.

We also notice that DTA performs slightly better than DIPM (by at most 2.5%). This is because DTA aims to achieve user equilibrium, with which shortening the travel time is the only objective. DIPM, on the other hand, aims to achieve system equilibrium and reduce both the travel time and traffic imbalance. Furthermore, as shown in Section VI-B, DIPM can work well in real-time. DTA, on the other hand, cannot, due to its very high computational complexity coupled with high traffic dynamics and imperfect traffic knowledge [12].

To investigate how travel times are affected by traffic congestion, we have selected three OD pairs with a reasonable large number of vehicles (at least 40 vehicles each) and examined the travel times of these vehicles. We show the travel times in Fig. 9, one sub-figure for each OD pair. We sort the vehicles based on their departure times, and use their ranks as their IDs in the figure, with vehicles departing earlier having smaller IDs. At the beginning of the simulation, there are no vehicles on the road network; then, vehicles enter and depart the network gradually. The first vehicles to enter the network experience lighter traffic.

As shown in Fig. 9, travel times increase for all algorithms when traffic becomes heavy. The travel times are the longest with SSP, because SSP cannot reroute vehicles to avoid congestion. EBkSP, AR*, and DTA can perform better than DIPM when traffic is light. However, their advantages cannot sustain when traffic becomes heavy; for vehicles with large IDs, their travel times are the lowest with DIPM. The reason is that these algorithms give more weight to shortening paths, instead of balancing traffic load in the network. When traffic is light and congestion is unlikely, they achieve better performance by making vehicles traveling shorter distances. But, when traffic is heavy, having vehicles moving along their shortest paths increases the chance of traffic congestion, and thus increases the travel times. DIPM, on the other hand, reduces traffic imbalance and works best for scenarios with medium to high congestion. These results can inform a real-life system to decide when to use DIPM.

Furthermore, we analyzed the number of reroutings produced by each algorithm. This is an important parameter because the drivers may not use the system if too they have to go through too many reroutings. Table V shows the mean value and standard deviation of the number of reroutings for all vehicles. SSP does

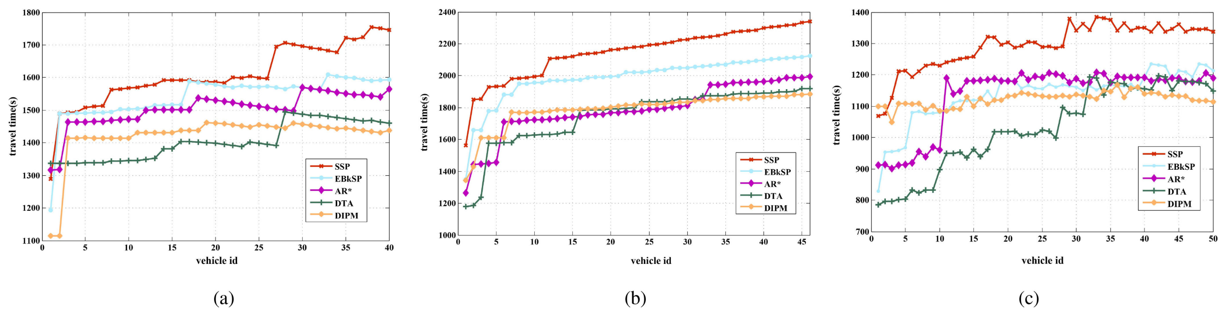


Fig. 9. Effect of congestion for vehicles with the same OD pairs. (a) Travel times for 40 cars with OD pair #1. (b) Travel times for 45 cars with OD pair #2. (c) Travel times for 50 cars with OD pair #3.

TABLE V
NUMBER OF REROUTINGS FOR ALL VEHICLES

metrics	EBkSP	AR*	DTA	DIPM
mean value	1.538	1.429	2.5	1.419
standard deviation	1.301	1.409	0.8	1.399

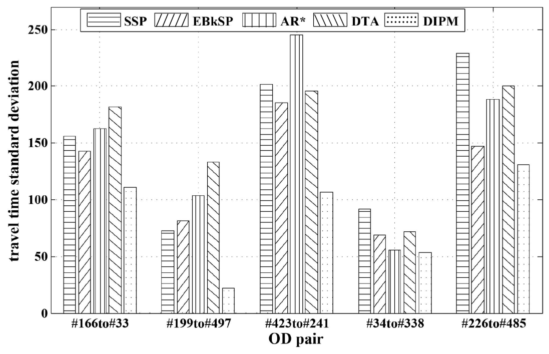


Fig. 10. Fairness comparison through travel time standard deviation.

not provide rerouting, so it is not shown in the table. The results show that DIPM performs the best, while DTA has the most reroutings. Since the number of reroutings is relatively low, we believe that DIPM can be acceptable in practice.

D. Fairness Comparison

As argued in Section I, driver fairness is very important for widespread system adoption. Fig. 10 shows a fairness comparison between DIPM and the other 4 algorithms. The metric used in this comparison is the standard deviation of the travel times for vehicles with the same OD pair (but different departure times). The results demonstrate that the standard deviation is the lowest for DIPM. This means that DIPM is fairer than the other algorithms: vehicles with the same OD pair have similar travel times. In addition to fairness, the results also emphasize that the drivers can expect predictable/stable travel times with DIPM. These benefits come from the combined objectives of DIPM of reducing both the travel times and the traffic imbalance in the network.

Aiming to analyze the traffic load balance quantitatively, Fig. 11 shows the average traffic load ratio in each time slot. The traffic load ratio is defined as $\frac{N_i}{c_i}$, where N_i denotes the number of vehicles on road segment r_i in this time slot and c_i is this road segment's capacity. Only segments that have vehicles are included in the the mean value's calculation. The higher the ratio is, the heavier the traffic load is in this time slot. The figure shows that the performances of traffic load ratio as SSP $>$ EBkSP $>$ AR* $>$ DTA $>$ DIPM. Compared with the other four algorithms, DIPM's traffic load ratio ranges from 0.6 to 0.7 in most time slots. According to the standard deviation, the traffic load is the most stable in DIPM across all time slots, which means that DIPM's traffic load balance is better than the load balance obtained by the other algorithms. The final question that we address in this evaluation is: what is the cost of fairness in terms of increased travel time for drivers when compared to DTA, which provides the optimal driver travel times? We have already seen in Section VI-C that the average travel time of DIPM is just slightly higher than that of DTA. Now, we compare the travel times for individual drivers between these two algorithms.

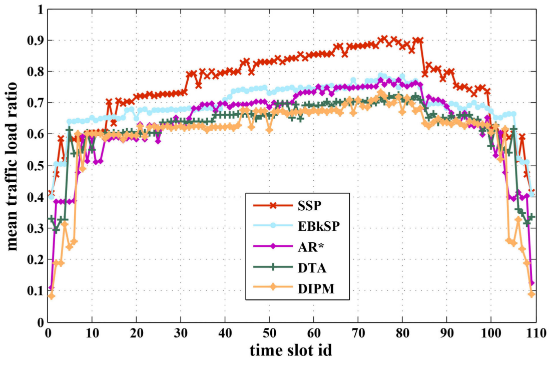


Fig. 11. Traffic load ratio in different time slots.

Specifically, we want to find out how many drivers benefit from DIPM and use less time to reach their destinations, and how many vehicles suffer longer travel times. For this purpose, we show the CDF of the relative travel times in Fig. 12. The relative travel time for a driver is the ratio between the travel time with DIPM and the travel time with DTA. Thus, a relative travel time below 1 means that the vehicle uses less time to reach the destination with DIPM.

Fig. 12 shows that about 70% of the vehicles have smaller travel times with DIPM than they do with DTA. This indicates that, compared to DTA, although DIPM cannot shorten the average travel time, it can shorten the travel times of most

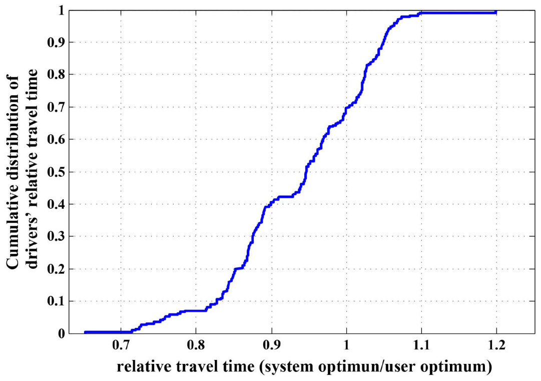


Fig. 12. CDF of relative travel time between DIPM and DTA.

vehicles. Another 30% of the vehicles have larger travel times with DIPM, but only a very small percentage (<3%) have relative travel times larger than 1.1, and the relative travel times of all vehicles are lower than 1.2. This is the cost of fairness. Most of these drivers are altruistic and choose to follow the alternative routes suggested by DIPM. This indicates that DIPM can achieve fairness for all drivers, better travel times for most drivers, and effectively limits the increase on travel times for altruistic drivers. Thus, DIPM can be a practical solution for traffic optimization.

VII. Conclusion

This paper has proposed a system-wide traffic optimization model that minimizes the sum of the total travel time and a time representation of the effects of traffic load imbalance in the road network. We proposed an analytic formulation for this optimization and an algorithm, DIPM, that solves the optimization. The experimental results have demonstrated that DIPM outperforms existing practical algorithms in terms of travel time. Furthermore, its travel time benefits become more apparent during traffic congestion, when most existing solutions do not perform well. DIPM also improves the driver fairness by providing similar travel times for drivers with the same OD pairs. Finally, our results show that DIPM can provide results in real-time. Its fairness and real-time features make DIPM practical for real-life traffic navigators. As future work, to ensure the scalability of our system, we will consider multiple cooperative servers. Then, we will study how to divide the road network among these servers and how to execute our algorithm in this distributed setting.

Appendix A
Proof of Theorem 1 on Solution Feasibility

Based on [17], an inequality constrained minimization model has a feasible solution if the model satisfies two requirements: 1) the objective and inequality constraints are twice continuously differentiable convex functions; and 2) For the matrix in the equality constraint A, $A \in \mathbb{R}^{y\times n}$, its rank should satisfy $rankA = y < n$.

In our optimization model, as described in Section V, all the functions in Eq. (14) take $N_i(T)$ as their variable after certain formula transformations. The objective function, $tte^i_m(T) + \alpha_i(T)\cdot B^i_m(T) + \beta^i_m(T)\cdot E^i_m(T)$, can be transformed into

$$f_0(N_i(T)) = \frac{\gamma_1 L_i}{v_i^{leg}(T)}\left[1 + k_1\left(\frac{N_i(T)}{c_i}\right)^{k_2}\right] \pm \frac{L_i}{v_i^{leg}(T)}\left[1 + k_3\left(\frac{N_i(T) - N_i^{thr}(T)}{c_i}\right)^{k_4}\right]. \tag{24}$$

In equation (24), γ_1 is a constant ratio; the positive or negative value of the second term depends on the value of $N_i(T)$. The twice differentiation of the function is continuous and convex:

$$\nabla^2 f_0(N_i(T)) = \frac{\gamma_1 L_i k_1 k_2(k_2-1)}{v_i^{leg}(T)\cdot(c_i)^2}\left(\frac{N_i(T)}{c_i}\right)^{k_2} \pm \frac{k_3 k_4 L_i(k_4-1)}{v_i^{leg}(T)\cdot(c_i)^2}\left(\frac{N_i(T) - N_i^{thr}(T)}{c_i}\right)^{k_4}. \tag{25}$$

Among the inequality constraints, the first inequality constraint, Eq. (2), can be simplified into $f_1(N_i(T)) = \gamma_2 N_i(T)$, $\forall i \le p$, utilizing the relationship described in Eq. (1). Since γ_2 is a ratio constant, $\nabla f_1(N_i(T)) = \gamma_2$, and $\nabla^2 f_1(N_i(T)) = 0$. Thus, the first inequality constraint also satisfies the first requirement.

The second inequality constraint described in Eq. (7) can be divided into two inequality functions with $N_i(T)$ as their variables, denoted $f_{2,1}$ and $f_{2,2}$. Here, $f_{2,1}$ is the lower bound and $f_{2,2}$ is the upper bound. Since the value of second constraint is between $\frac{N_i(T)}{2}$ and $N_i(T)$, both of $f_{2,1}$ and $f_{2,2}$ are linear functions. Specifically, $\nabla f_{2,1}(N_i(T)) = -\frac{1}{2}$, $\nabla^2 f_{2,1}(N_i(T)) = 0$, $\nabla f_{2,2}(N_i(T)) = 1$, and $\nabla^2 f_{2,2}(N_i(T)) = 0$. Thus, the second inequality constraint satisfies the first requirement.

The last inequality constraint is described in Eq. (8), i.e., for $\forall i \le p$, $f_3(N_i(T)) = E^i_m(T) - \theta tte^{opt}_m(T) \le 0$. The model uses a ratio γ_3 to describe the difference between $tte^i_m(T)$ and $tte^{opt}_m(T)$, i.e., $tte^{opt}_m(T) = \gamma_3 tte^i_m(T)$, where $\gamma_3 \le 1$. Thus, the last inequality constraint f_3 can be rewritten as:

$$f_3(N_i(T)) = [1 - \gamma_3(1+\theta)]tte^i_m(T) = [1 - \gamma_3(1+\theta)]\frac{L_i}{v_i^{leg}(T)}\left[1 + k_1\left(\frac{N_i(T)}{c_i}\right)^{k_2}\right]. \tag{26}$$

The twice differentiation of f_3 shown below is continuous and convex. Thus, it also satisfies the first requirement.

$$\nabla^2 f_3(N_i(T)) = [1 - \gamma_3(1+\theta)]\frac{L_i k_1 k_2(k_2-1)}{(c_i)^2 v_i^{leg}(T)}\left(\frac{N_i(T)}{c_i}\right)^{k_2}. \tag{27}$$

The matrix of the equality constraint in the proposed model is $N = \{N_1(T), \ldots, N_i(T), \ldots N_p(T)\}^{\mathrm{T}}$ with a rank of 1. Thus, $rankN = 1 < p$, where p denotes the number of road segments. This means that the matrix satisfies the second requirement.

To summarize, the optimization model satisfies both requirements. Thus, it has a feasible solution, and the optimal result is attained.

APPENDIX B
PROOF OF THEOREM 2 ON DIPM CONVERGENCE

Appendix A has proven that $f_0(x)$, $f_1(x)$, $f_2(x)$, and $f_3(x)$ are continuous twice differentiable functions.

For Eq. (19), assuming $x^*(t)$ is the optimal result of x, when $x = x^*(t)$, there exists a w that satisfies the KKT conditions:

$$t\nabla f_0(x) + \sum_{a=1}^{3} \frac{-1}{f_a(x)} \nabla f_a(x) + A^T w = 0,\ Ax = b. \quad (28)$$

Eq. (28) can be considered as a Lagrange function of Eq. 19. It reaches its minimum value when $x = x^*(t)$. The function can be further rewritten as:

$$\mathscr{L}[x, \zeta_a^*(t), \nu_a^*(t)] = f_0(x) + \sum_{a=1}^{3} \zeta_a^*(t) f_a(x) + \nu_a^*(t)(Ax - b) \quad (29)$$

In Eq. (29), $\mathscr{L}[x, \zeta_a^*(t), \nu_a^*(t)]$ is the Lagrange function; $\zeta_a^*(t) = \frac{1}{-tf_a(x)}$; and $\nu_a^*(t) = w/t$. The function reflects the approximation of the original model described in Eq. (14). Thus, the original optimal result *opt** is greater than the optimal result of our method, i.e., the minimal value of Lagrange function $f_0(x^*(t))$:

$$\begin{aligned} opt^* &\geq \mathscr{L}[x^*(t), \zeta_a^*(t), \nu_a^*(t)] \\ &= f_0(x^*(t)) + \sum_{a=1}^{3} (\zeta_a^*(t) f_a(x^*(t))) + \nu_a^*(t)(Ax^*(t) - b) \\ &= f_0(x^*(t)) + \sum_{a=1}^{3} \left(\frac{f_a(x^*(t))}{-tf_a(x^*(t))} \right) + w/t(Ax - b) \\ &= f_0(x^*(t)) + \sum_{a=1}^{3} \left(\frac{1}{-t} \right). \end{aligned} \quad (30)$$

Based on Eq. (30), $f_0(x^*(t)) \leq opt^* + \frac{3}{t}$. This indicates that the optimal result of our method $f_0(x^*(t))$ will approach the original optimal result *opt** when $t \to \infty$. Therefore, DIPM converges.

APPENDIX C
PROOF OF THEOREM 3 ON DIPM COMPUTATION COMPLEXITY

The algorithm includes four steps in each time slot: 1) load the traffic data of the current time slot; 2) determine optimal routes and altruistic routes for all vehicles; 3) calculate an optimal altruistic ratio with DIPM; and 4) update the traffic data to be used in the next time slot. Among these steps, the second and third steps are the main parts, and determine the complexity of the algorithm.

The second step is finished using an enhanced version of the Yen's algorithm [19]. The enhancement is to estimate travel times more accurately by taking into consideration traffic density, and does not increase complexity. The complexity of Yen's algorithm depends on the shortest path algorithm used to compute the spur paths. Let us assume that Dijkstra's algorithm is used. With a Fibonacci heap, the complexity of Dijkstra's algorithm can be reduced to $O(p + N \log N)$ [19], where N is the number of road intersection and p is the number of road segments. The Yen's algorithm makes KN calls to the Dijkstra's algorithm when computing the spur paths, where K is the number of paths provided for vehicles, with a default value of 2 in our experiments. Thus, the complexity of the second step is $O(KN(p + N \log N))$. This shows that the complexity of step 2 is mainly determined by the number of the road intersections.

The complexity of step 3 is mainly determined by the number of iterations. Based on the self-concordance of barrier function [20], the complexity of IPM algorithm is $O(\sqrt{n} \log(\frac{n}{\epsilon}))$, where n is the number of inequality constraints in the model (3 in our algorithm), ϵ is the default value of error.

The DIPM algorithm is called in each time slot to compute the optimal rerouting for the set of OD pairs. s is cardinality of the set of OD pairs for all drivers, where the origin for each driver is considered its current segment. Thus, the complexity of step 3 is $O(s + \sqrt{n} \log(\frac{n}{\epsilon}))$.

The overall computational complexity of DIPM algorithm is $O(KN(p + N \log N) + s + \sqrt{n} \log(\frac{n}{\epsilon}))$, which is determined by the size of road network and the number of OD pairs in each time slot.

The DIPM complexity is reduced in two ways: (1) The barrier function parameters' calculation. DIPM need not calculate A in every time slot. Although b is different from IPM, it still needs calculation in the barrier function. Suppose that the number of calculation parameters is ϱ in IPM, therefore it will be $\varrho - 1$ in DIPM. In each time slot, DIPM has one less parameter (i.e., A) calculation; the reduction will be $(\sqrt{n} \log(\frac{n}{\epsilon}))$. (2) The number of time slots. Because DIPM decreases the time for parameters' calculation, it may end the progress earlier than IPM. Let us assume that the number of time slots DIPM decreases is Δz. Therefore, comparing to IPM over all time slots, the reduction of DIPM in complexity is $\Delta z(\sqrt{n} \log(\frac{n}{\epsilon}))$.

REFERENCES

[1] T. Mao *et al.*, "Aic2018 report: Traffic surveillance research," in *Proc. Conf. Comput. Vision Pattern Recognit. Workshop*, 2018, pp. 82–92.

[2] *Global Mobility Report 2017: Tracking Sector Performance*, Washington DC, License: Creative Commons Attribution CC BY 3.0, 2017.

[3] P. J. Owens, B. Kean, and G. Apostolopoulos, "Optimizing traffic load in a communications network," U.S. Patent No. 9648133. Washington, DC, May 9, 2017.

[4] F. Cunha *et al.*, "Data communication in VANETs: Protocols, applications and challenges," *Ad Hoc Netw.*, vol. 44, pp. 90–103, 2016.

[5] J. Wan, J. Liu, Z. Shao, A. V. Vasilakos, M. Imran, and K. Zhou, "Mobile crowd sensing for traffic prediction in internet of vehicles," *Sensors*, vol. 16, no. 1, pp. 1–15, 2016.

[6] M. M. Rahman, J. R. Mou, K. Tara, and M. I. Sarkar, "Real time Google map and Arduino based vehicle tracking system," in *Proc. IEEE Int. Conf. Elect., Comput. Telecommun. Eng.*, Dec. 2016, pp. 1–4.

[7] S. Hu, L. Su, H. Liu, H. Wang, and T. F. Abdelzaher, "Smartroad: Smartphone-based crowd sensing for traffic regulator detection and identification," *ACM Trans. Sensor Netw.*, vol. 11, no. 4, pp. 1–27, 2015.

[8] M. Wang, H. Shan, R. Lu, R. Zhang, X. Shen, and F. Bai, "Real-time path planning based on hybrid-VANET-enhanced transportation system," *IEEE Trans. Veh. Technol.*, vol. 64, no. 5, pp. 1664–1678, May 2015.

[9] S. Wang, S. Djahel, Z. Zhang, and J. McManis, "Next road rerouting: A multiagent system for mitigating unexpected urban traffic congestion," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 10, pp. 2888–2899, Oct. 2016.

[10] C. Guo, D. Li, G. Zhang, and M. Zhai, "Real-time path planning in urban area via VANET-assisted traffic information sharing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 5635–5649, Jul. 2018.

[11] J. Jeong, H. Jeong, E. Lee, T. Oh, and D. H. Du, "SAINT: Self-adaptive interactive navigation tool for cloud-based vehicular traffic optimization," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 4053–4067, Jun. 2016.

[12] J. Pan, I. S. Popa, K. Zeitouni, and C. Borcea, “Proactive vehicular traffic rerouting for lower travel time,” *IEEE Trans. Veh. Technol.*, vol. 62, no. 8, pp. 3551–3568, Oct. 2013.

[13] Z. Zhou, B. De Schutter, S. Lin, and Y. Xi, “Two-level hierarchical model-based predictive control for large-scale urban traffic networks,” *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 2, pp. 496–508, Mar. 2017.

[14] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, “Improving the efficiency of stochastic vehicle routing: A partial lagrange multiplier method,” *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3993–4005, Jun. 2016.

[15] H. Guo, Z. Cao, M. Seshadri, J. Zhang, D. Niyato, and U. Fastenrath, “Routing multiple vehicles cooperatively: Minimizing road network breakdown probability,” *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 1, no. 2, pp. 112–124, Mar. 2017.

[16] S. Alexander, and R. Dowling, “Improved speed-flow relationships for planning applications,” *Transp. Res. Rec.: J. Transp. Res. Board*, vol. 1572, pp. 18–23, 1997.

[17] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[18] Kheirfam and Behrouz, “A full-Newton step infeasible interior-point method based on a new search direction,” *Pacific J. Optim.*, vol. 13, no. 3, pp. 463–473, 2017.

[19] M. L. Fredman and R. E. Tarjan, “Fibonacci heaps and their uses in improved network optimization algorithms,” *J. ACM*, vol. 34, no. 3, pp. 596–615, 1987.

[20] D. D. Hertog, *Interior Point Approach to Linear, Quadratic and Convex Programming: Algorithms and Complexity*, vol. 277. Berlin, Germany: Springer, 2012.

[21] B. N. Janson, “Dynamic traffic assignment for urban road networks,” *Transp. Res. B: Methodological*, vol. 25, nos. 2/3, pp. 143–161, 1991.

[22] T. V. Mathew and K. K. Rao, *Fundamental Relations of Traffic Flow.* Lecture Notes in Transportation Systems Engineering, India: Department of Civil Engineering Indian Institute of Technology Bombay, 2017.

[23] F. Ramm, J. Topf, and S. Chilton, *OpenStreetMap: Using and Enhancing the Free Map of the World.* Cambridge, U.K.: UIT Cambridge, 2011.

[24] S. Kasapovic and L. B.-Mehmedovic, “Simulation VANET networks on a random and realistic spatial scenario,” in *Proc. Int. Conf. Appl. Phys., Syst. Sci. Comput.*, 2017, pp. 245–251.

[25] E. D. M.-Hooks and H. S. Mahmassani, “Least expected time paths in stochastic, time-varying transportation networks,” *Transp. Sci.*, vol. 34, no. 2, pp. 198–215, 2000.

[26] Z. Cao, S. Jiang, J. Zhang, and H. Guo, “A unified framework for vehicle rerouting and traffic light control to reduce traffic congestion,” *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 7, pp. 1958–1973, Jul. 2017.

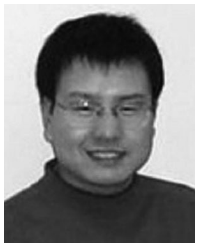
[27] Z. Cao, H. Guo, and J. Zhang, “A multiagent-based approach for vehicle routing by considering both arriving on time and total travel time,” *ACM Trans. Intell. Syst. Technol.*, vol. 9, no. 3, pp. 25, 2018.

[28] E. Q. V. Martins and M. B. P. Marta, “A new implementation of Yen’s ranking loopless paths algorithm,” *Quart. J. Belgian, French Italian Operations Res. Soc.*, vol. 1, pp. 121–133, 2003.

Chang Guo received the B.S. degree in communication engineering from Donghua University, Shanghai, China, in 2014 and is in successive postgraduate and doctoral program of study to pursue the doctor’s degree since 2015 with Donghua University. Her research interests focus on solving the traffic congestion problem based on vehicular ad hoc networks, which include real-time traffic information acquisition and sharing, data delivery delay in VANETs, dynamic traffic condition prediction, dynamic path planning, and traffic load balance in road network.

Demin Li received the Ph.D. degree in electronic and computer engineering from the Nanjing University of Science and Technology, Nanjing, China, in 1998. He is currently a Professor with the Department of Communication Engineering, College of Information Science and Technology, Donghua University, Shanghai, China. His recent research interests include telecommunication system engineering, wireless mobile networking, mobile decision theory and mobile decision support systems. He is currently as Associate Chairman of the Circuits and Systems Committee in Shanghai.

Guanglin Zhang received the Ph.D. degree in information and communication engineering from Shanghai Jiao Tong University, Shanghai, China, in 2012. From 2013 to 2014, he was a Postdoctoral Research Associate with the Institute of Network Coding, Chinese University of Hong Kong. From 2013 to 2017, he was an Associate Professor with the Department of Communication Engineering, Donghua University, Shanghai, China, where he is currently a Professor and the Department Chair with the Department of Communication Engineering. His research interests include capacity scaling of wireless networks, vehicular networks, smart microgrids, and mobile edge computing. He is a Technical Program Committee Member for the IEEE Global Communications Conference (ICC) 2016–2018, the IEEE International Conference on Communications (GC) 2014–2018, the IEEE ICDCS 2019, IEEE WCNC 2019, the IEEE Vehicular Technology Conference in Fall 2017, the IEEE/CIC International Conference on Communications in China in 2014, and the International Conference on Wireless Communications and Signal Processing in 2014, the Asia-Pacific Conference on Communications in 2013, and the International Conference on Wireless Algorithms, Systems, and Applications in 2012. He serves as the Local Arrangement Chair of ACM TURC 2017, 2019, and the Vice TPC Co-Chair of ACM TURC 2018, 2020. He is currently the Local Arrangement Chair of ACM Mobihoc 2020. Hewas the recipient of the ACM China Outstanding Service Award 2017, 2018, 2019. He is an Associate Editor on the Editorial Board of China Communications.



Xiaoning Ding received the Ph.D. degree in computer science and engineering from the Ohio State University, Columbus, OH, USA. He is an Associate Professor with the New Jersey Institute of Technology, Newark, NJ, USA His interests are in the area of experimental computer systems, such as distributed systems, virtualization, operating systems, and storage systems.

Reza Curtmola received the Ph.D. degree in computer science from The Johns Hopkins University, Baltimore, MD, USA. He is a Professor of computer science with the New Jersey Institute of Technology, Newark, NJ, USA. He has authored or coauthored more than 65 papers under the umbrella of cybersecurity. His research focuses on the security of cloud services, security of the software supply chain, and applied cryptography. He is the recipient of the NSF CAREER Award and has participated in several other projects funded by NSF and DARPA.

Cristian Borcea (Member, IEEE) received the Ph.D. degree from Rutgers University, Newark, NJ, USA, in 2004. He is currently a Professor with the Department of Computer Science, New Jersey Institute of Technology, Newark, NJ, USA. He is also a Visiting Professor with the National Institute of Informatics, Tokyo, Japan. His research interests include mobile computing and sensing; vehicular computing and networks; cloud and distributed systems, and computational advertising. He is a member of the ACM.